instruction of a process, wherein the architectural definition of the instruction does not call for an interrupt, the interrupt criteria being [synchronously] based at least in part on the table entry associated with [a memory state of the computer and] the address of the instruction, [the architectural definition of the instruction not calling for an interrupt,] the interrupt circuitry being designed to invoke a handler for the interrupt, the handler being responsive to a content [the contents] of the table entry to affect the instruction pipeline circuitry to effect control of an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on the contents of a table entry associated with the address range in which the instruction lies.

2. (twice amended) A method, comprising the steps of:

as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, consulting a table, the table being indexed [addressed] by the address of instructions executed, entries of the table containing [for] attributes of [the] instructions whose addresses index to the respective entries; and

controlling an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on a content [the contents] of a table entry associated with the address of the instruction.

3. (amended) The method of claim 2, wherein the control of control transfer behavior includes transfer of execution control to a second [different] instruction for execution.

4. (amended) The method of claim 3, wherein the second [different] instruction is coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

7. (amended) The method of claim 2, wherein:

[wherein] entries [each entry] of the table describe[s] a likelihood of the existence of an alternate coding of instructions located in [the] respective corresponding address ranges.

D

8. (amended) The method of claim 2, wherein:

[wherein] entries [each entry] of the table correspond[s] to [a] pages managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is [a table entry being] integrated with virtual memory address translation circuitry of the computer.

9. (amended) The method of claim 2, further comprising the steps of:

synchronously triggering an interrupt on execution of an instruction of a process in accordance with interrupt criteria, the interrupt criteria being [synchronously] based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call [not calling] for an interrupt.

11. (amended) The microprocessor chip of claim 10, further comprising:

a binary translator programmed to translate at least a selected portion of a computer program from a first binary representation to a second binary representation; and

within the pipeline control circuitry is further designed to initiate a determination of whether to transfer control from an execution of the instruction, the instruction being an instruction of the first binary representation of the program, to the second binary representation, and effective to initiate the determination with neither a query nor a transfer of control to the second binary representation being coded into the first binary representation.

12. (amended) The microprocessor chip of claim 10, further comprising:

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of the [an] instruction [of a process] in accordance with interrupt criteria, the interrupt criteria being [synchronously] based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call [not calling] for an interrupt.

D

13. (amended) The microprocessor chip of claim 12, further comprising:

interrupt handler software designed to service the interrupt and to effect the altering of a transfer of control behavior, the altering being in the form of returning control to an instruction flow of the process other than the instruction flow triggering the interrupt, the returned-to instruction flow for carrying on non-error handling normal processing of the process.

14. (amended) A microprocessor chip, comprising:

instruction pipeline circuitry;

address translation circuitry; and

a lookup structure having entries [an entry] associated with [each] corresponding address ranges generated by the instruction pipeline circuitry and translated by the address translation circuitry, the entries [entry] describing a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range.

18. (amended) The microprocessor chip of claim 14, further comprising:

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of an instruction of a process, synchronously based on interrupt criteria, the interrupt criteria based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call [not calling] for an interrupt.

19. (amended) A microprocessor chip, comprising:

instruction pipeline circuitry; and

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of an instruction of a process in accordance with interrupt criteria, the interrupt criteria being [synchronously] based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call [not calling] for an interrupt.

22. (amended) The microprocessor chip of claim 20, wherein the [instruction text beginning at the] returned-to instruction flow is logically equivalent to the instruction flow [text] beginning at the interrupted instruction.

24. (amended) A method, comprising the steps of:

as an integral part of processing an instruction in instruction pipeline circuitry of a computer, consulting a lookup structure of entries, entries of the lookup structure [each entry] corresponding to [an] address ranges translated by address translation circuitry, and the entries describe [entry describing] a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address ranges.

26. (amended) The method of claim [16.] 24, further comprising the step of:

altering a behavior of the instruction in a manner incompatible with the architectural definition in an emulated architecture of the instruction, based on a content [the contents] of the entry corresponding to the address range containing the instruction.

27. (amended) The method of claim 24,

wherein each entry corresponds to a page managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is [a table entry being] integrated with virtual memory address translation circuitry of the computer.

28. (amended) The method of claim 24, further comprising the step of:

based on a content [the contents] of the entry, transferring control to the alternative coding, the alternative coding being an instruction flow of the process other than the instruction flow triggering the consulting, the returned-to instruction flow being programmed to carry on non-error handling normal processing of the process.

29. (amended) The method of claim 24, further comprising the step of:

based on a content [the contents] of the entry, synchronously triggering an interrupt in accordance with interrupt criteria, the interrupt criteria being based at least in part on a

D

memory state of the computer and the address of the instruction, wherein the architectural definition in an emulated architecture of the instruction does not call for an interrupt.

30. (amended) A method, comprising the steps of:

on execution of an instruction of a process, [the architectural definition of the instruction not calling for an interrupt,] synchronously triggering an interrupt based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction does not call for an interrupt.

34. (amended) The microprocessor chip of claim 30, further comprising:

table lookup circuitry designed to index into a table by a memory address of a memory reference arising during execution of an instruction, and to retrieve a table entry corresponding to the address;

the instruction pipeline circuitry being responsive to a content [the contents] of the table to affect a manipulation of data or transfer of control defined for the instruction.

Please add the following new claims:

36. (new) The microprocessor chip of claim 10, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

37. (new) The microprocessor chip of claim 19, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

38. (new) The method chip of claim 30, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

39. (new) A method, comprising the steps of:

as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, retrieving an entry from a table, the entry of the table being indexed by the address of a memory reference arising during execution of the instruction, the table entry being distinct from the memory referenced by the memory reference;

based on a content of the table entry, altering a manipulation of data or transfer of control behavior of the instruction in a manner incompatible with the architectural definition in an emulated architecture of the instruction.

40. (new) The method of claim 39, wherein the entries of the table correspond to ranges of memory addresses.

41. (new) The method of claim 39, wherein the entry of the table is indexed by the address of instructions executed.

42. (new) The method of claim 39, wherein:

entries of the table correspond to address ranges, and the entries describe a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range.

43. (new) The method of claim 39:

wherein the architectural definition of the instruction in the emulated architecture does not call for an interrupt;

and further comprising the step of synchronously triggering an interrupt based at least in part on a memory state of the computer and the address of the instruction.

44. (new) The method of claim 39, wherein the control of transfer of control behavior includes transfer of execution control to a second instruction for execution.

45. (new) The method of claim 44, wherein the second instruction is coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

46. (new) The method of claim 39, wherein the control of architecturally-visible data manipulation behavior includes changing an instruction set architecture under which instructions are interpreted by the computer.

47. (new) The method of claim 39, wherein the behavior control includes selecting between two different instruction set architectures, and the computer includes instruction pipeline circuitry designed to effect interpretation of computer instructions under the two instruction set architectures alternately.

48. (new) The method of claim 39:

wherein each entry of the table corresponds to a page managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is integrated with virtual memory address translation circuitry of the computer.

49. (new) The method of claim 39, further comprising the steps of:

triggering an interrupt on execution of an instruction of a process, synchronously based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

50. (new) An apparatus, comprising:

instruction pipeline circuitry; and

table lookup circuitry designed to retrieve a table entry from a table indexed by an address of an instruction fetched for execution;

the instruction pipeline circuitry being responsive to a content of the table entry to control an architecturally-visible data manipulation behavior or control transfer behavior of the fetched instruction based on a content of the table entry associated with the address of the instruction.